REMARKS

Claims 1 to 35 have been canceled without prejudice. Claims 36 to 68 have been added and are pending. No new matter has been added. Claims 1 to 35 were rejected. Applicant thanks the Examiner for withdrawing the previous indefiniteness rejection. Applicant requests reconsideration and allowance of the present application for the following reasons.

Rejections under 35 U.S.C. § 103(a)

As an initial matter, claims 1 to 35 have been canceled, rendering the outstanding rejections moot. However, to the extent the new claims include features analogous to the canceled claims, Applicant will outline why the cited prior art does not read on the present invention.

Claims 36 to 47:

For example, claim 36 recites in whole:

> A method for monitoring updates in a software repository in a **multi-author software design environment**, comprising:
> constructing a first snapshot of a set of **software module source code** stored by the software repository at a first point in time, the source code represented **by a plurality of objects**;
> constructing a second snapshot of the set of software module source code stored by the software repository at a second point in time;
> comparing the first snapshot with the second snapshot;
> rating each detected difference according to a backward compatibility metric, the backward compatibility metric representing a probability that the detected difference renders an attribute of the second snapshot incompatible with a similar attribute of the first snapshot;
> determining an overall backward compatibility score for the second snapshot, based on the rated differences; and
> issuing an alert message **to registered authors of the set of software module source code** when the overall backward compatibility exceeds a backward compatibility threshold.

U.S. Patent No. 6,986,132 B1 ("Schwabe") was cited as disclosing the API comparison between two versions of a binary file. However, Schwabe is an end-user application that verifies executing binaries. Schwabe has nothing to do with "a multi-author software design environment," snapshots of "source code . . . represented by a plurality of objects," nor "alert message[s] to registered authors of the set of software module source code." For example, col. 5, lines 6 to 21 outline the basic version of the Schwabe method, where the calling

context is verified, the binaries are verified, and the virtual stack is verified/updated. All of these tasks are exclusive to program execution and fail to disclose multiple authors and verified *source code*.

Schwabe also does not disclose "constructing a first [or second] snapshot of a set of software module[s]." The Office believes this feature would be obvious, despite providing no prior art disclosing the feature. On page 3 of the present Office Action, a snapshot is characterized as a copy of an API, which is held obvious without explicit support. Even if this was an accurate characterization, it requires one to assume the API belongs to a compiled/completed program. The Office states that "applications for taking snapshots of a file are well known in the field." The applications known in the field relate to constructing an API of a compiled binary, as discussed in Schwabe. However, this is far simpler and very different from constructing a snapshot of a shared source code repository, storing source code still under construction.

Since Schwabe also does not disclose "rating each detected difference according to a backward compatibility metric, . . . [and] determining an overall backward compatibility score for the second snapshot, based on the rated differences," U.S. Patent No. 7,069,474 B2 ("Atallah") is cited as disclosing this feature. However, like Schwabe, Atallah discloses an end-user system related to compiled and finished applications. Atallah discloses, "systems and methods for assessing the risk of **binary compatibility failure** between software modules, or binary files thereof, and an application binary interface." Atallah at col. 1, lines 63 to 66 (emphasis added). "The term binary compatibility refers to maintaining compatibility between a binary (**i.e., compiled**) version of a first software module, e.g., an application module, and a second software module, e.g., *an operating system*." Atallah at col. 3, lines 6 to 9 (emphasis added). Therefore, even if Atallah does disclose compatibility tests, it is in a much different context. Atallah, dealing exclusively with binary (i.e., compiled) versions of software and a functioning second software module, such as an operating system (e.g., the Sun Solaris system Atallah relates to), does not disclose a "**multi-author software design environment**," or "a set of software module **source code** that **defines a plurality of objects.**"

Since Schwabe and Atallah are wholly unrelated to a "**multi-author software design environment,**" *Lin et al.*, "Multiuser Collaborative Work in Virtual Environment based CASE Tool" ("Lin") has been cited. Lin discloses a "Fine-grained locking mechanism[, which] moves the contention from the level of [an] object to the level of single attribute[s] in

[an] object, thus chang[ing] the nature of access contentions." Lin at § 3.2.1. As explained in section 3.2 and its sub-sections, for systems that require read/write locking (e.g. a single shared data file with multiple distributed authors/editors), Lin has moved the object level locking down to the attribute level (e.g., variable, method, etc.). Thus, more than one author/editor may concurrently modify the same object, but not the same attribute. This, other than possibly being a "multi-author software design environment," has nothing to do with the above quoted claim. Therefore, the combination of Schwabe and Atallah, with Lin is not proper for at least two reasons.

First, Lin would simply not function in combination with Schwabe or Atallah. Schwabe and Atallah are directed toward the compatibility of compiled executables, which require no authoring at all. Thus, it is not entirely clear how this could be combined with a software authoring environment at all. However, at the very least, it may be observed that Atallah is cited for the feature of "rating each detected difference according to a backward compatibility metric; [and] determining an overall backward compatibility score for the second version, based on the detected differences." However, this "rating each detected difference" is fundamentally inapplicable to Lin. Lin is directed to a "locking mechanism," the whole point of which is to eliminate all potential differences. In order for Lin to detect a difference in the relevant program, it would have to completely fail at performing its designed task. Thus, these references are simply not combinable, as one would render inoperable the other.

Second, related to the fact that the combination is incapable of functioning, there is simply no reason to combine these references. Schwabe and Atallah being directed to compiled programs, and checking compatibility on the existing system of the end-user. While Lin is directed toward preventing a code author from editing a source code attribute concurrently with any other code author. Lin has nothing to do with compatibility, existing systems, end-users, compiled code, and especially detecting a difference, which is the complete opposite of Lin's designed function. Thus, a person of ordinary skill in the art would have no reason to combine these references.

For at least the above mentioned reasons, the combination of Schwabe, Atallah, and Lin would not render claim 36 unpatentable, and Applicant respectfully requests the rejection be withdrawn. Claims 37 to 47 depend on claim 36 and should therefore be allowed for at least the same reasons.

Claims 48 to 59:

Claims 48 recites in whole:

> A system for monitoring updates in a software repository in **a multi-author software design environment**, comprising:
> a processor configured to construct a first snapshot of a set of software module **source code that defines a plurality of objects**;
> the processor configured to subsequently construct a second snapshot of the set of software module source code;
> the processor configured to compare the first snapshot with the second snapshot;
> the processor configured to rate each detected difference according to a backward compatibility metric;
> the processor configured to determine an overall backward compatibility score for the second version, based on the detected differences;
> the processor, in connection with an output device, configured to **issue an alert message to registered authors of the set of software module source code**.

Schwabe, Atallah, and Lin do not disclose the features of claim 48, and the combination of Lin with either of the other two references is not proper. Since Atallah and Schwabe are directed to detecting differences, presupposing at least the expectation of a difference occurring, and Lin is directed to completely preventing differences, the combination would be inoperable, as these are mutually exclusive functions. Further, since Atallah and Schwabe deal with the compatibility of compiled end-user programs, and Lin deals with a multi-user code authoring system, there would be no reason for a person of ordinary skill in the art to combine these very different references. For at least these reasons, claim 48 should be allowed. Claims 59 to 59 depend from claim 48 and should therefore be allowed for at least the same reasons.

Claims 60 to 69:

For example, claim 60 recites in whole:

> A computer-readable storage medium encoded with instructions configured to be executed by a processor, the instructions which, when executed by the processor, cause the performance of a method for monitoring updates in a software repository in a **multi-author software design environment**, comprising:
> constructing a first snapshot of a set of software module **source code** that **defines a plurality of objects**;
> subsequently, constructing a second snapshot of the set of software module source code;

> comparing the first snapshot with the second snapshot;
> rating each detected difference according to a backward compatibility metric;
> determining an overall backward compatibility score for the second version, based on the detected differences;
> issuing an alert message **to registered authors of the set of software module source code**.

Schwabe, Atallah, and Lin do not disclose the features of claim 48, and the combination of Lin with either of the other two references is not proper. Since Atallah and Schwabe are directed to detecting differences, presupposing at least the expectation of a difference occurring, and Lin is directed to completely preventing differences, the combination would be inoperable, as these are mutually exclusive functions. Further, since Atallah and Schwabe deal with the compatibility of compiled end-user programs, and Lin deals with a multi-user code authoring system, there would be no reason for a person of ordinary skill in the art to combine these very different references. For at least these reasons, claim 48 should be allowed. Claims 59 to 69 depend from claim 48 and should therefore be allowed for at least the same reasons.

CONCLUSION

In view of all of the above, it is believed that the rejections have been obviated or otherwise rendered moot, and that claims 36 to 68 are allowable. It is therefore respectfully requested that any outstanding rejections be withdrawn, and that the present application issue as early as possible.

If for any reason the Examiner believes that contact with Applicant's attorney would advance the prosecution of this application, he or she is invited to contact the undersigned at the number given below.

Although not believed necessary, the Office is hereby authorized to charge any fees required under 37 C.F.R. § 1.16 or § 1.17 or credit any overpayments to Deposit Account No. 11-0600.

Respectfully submitted,

Dated: March 13, 2009

By: /John B. Gillick/
John B. Gillick, Jr.
Registration No. 63,027

KENYON & KENYON LLP
One Broadway
New York, NY 10004
(212) 425-7200 (phone)
(212) 425-5288 (fax)
CUSTOMER NO. 53000